audiokinetic

Cube Integration

Cube Integration

Wwise

Table of Contents

Chapter 1. Welcome to the Wwise Cube Integration

Audiokinetic integrated the Wwise sound engine into the first person shooter game Cube. This version of Cube was provided to help you better understand how to integrate the Wwise sound engine into your game and provides a functional game for sound designers to experiment with Wwise. This game was chosen as an example for the simplicity of its structure, and because it is relatively easy to modify, if needed.

Using the Sound Engine

The project provides you with information on how the Wwise sound engine has been integrated into Cube. This Help document is divided into two main sections to provide you with comprehensive information about the integration of the Wwise sound engine into Cube.

- **Understanding Wwise's Sound Engine Integration into Cube** - Shows you how the Wwise sound engine was integrated in Cube.
- **The Wwise Project** - Details important components of the Wwise project and shows how to connect Wwise to the game, modify sound properties live, and integrate these changes using newly generated SoundBanks.

The Audiokinetic Cube Package

The Audiokinetic Cube Package includes the following components:

- Debug, profile, and release versions of Audiokinetic's Cube game
- A Wwise project that was used to modify and add sounds to the game
- The Cube game engine source code and documentation

Running the Cube Demo

With the advent of the Wwise Launcher, installing the Cube Demo sample will list it under the Samples tab where you can directly click **Run Cube** to launch the demo.

For Mac®, however, there are a couple of extra steps to follow beforehand.

Running the Cube Demo on Mac®

The Cube Demo for Mac requires libSDL2 to be installed. Without this, attempts to run the demo from the Launcher will present an error message: "*The SDL2 library is not detected. The Cube Demo for Mac requires SDL2 to be installed according to the instructions at https://www.audiokinetic.com/*

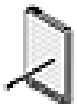
library/<version>/?source=Cube&id=install_libSDL2_for_mac. Please restart the Wwise Launcher after the installation is complete." Likewise, attempts to run the game from the Terminal will display this message: "*Library not loaded: @rpath / SDL2.framework / Versions /A / SDL2.*"

To install libSDL2 for Mac, do the following:

1. Download the latest libSDL2 Mac OS X Runtime Binaries package: https://www.libsdl.org/download-2.0.php .
2. Follow the instructions provided in the libSDL2 README file.

Running the Cube Demo in Other Ways

For different reasons, it may be that users prefer not to use the Launcher to run the Cube Demo. There are alternative methods to run the Cube Demo, with some differences depending on whether it is run on Mac or Windows.



Note

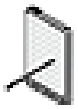
Before starting on Mac OS X, make sure the Wwise SDK for Mac is installed on the machine.

On Mac OS X

1. Method 1: Run under Xcode.
   a. Check that Xcode is installed. If Xcode is not present in the Applications folder, download and install it from the Mac App Store.
   b. Make sure the unzipped Cube demo folder shares the same parent folder as the installed Wwise SDK Mac distribution. For example, if the Wwise SDK folder is ${HOME}/Wwise/wwise_${version_tag}/SDK, then the Cube demo root folder must be ${HOME}/Wwise/wwise_${version_tag}/CubeDemo.
   c. In Xcode, open the following Xcode project: /path/to/CubeDemo/cube_source/src/Mac/CubeMac.xcodeproj.
   d. Build and run the project.
2. Method 2: Use the shell scripts.
   a. Make sure the unzipped Cube demo folder shares the same parent folder as the installed Wwise SDK Mac distribution. For example, if the Wwise SDK folder is ${HOME}/Wwise/wwise_${version_tag}/SDK, then the Cube demo root folder must be ${HOME}/Wwise/wwise_${version_tag}/CubeDemo.
   b. Locate the executable shell script for the desired configuration: /path/to/CubeDemo/Mac/<configuration>/bin/RunCubeDemo.sh.

c. If the .sh files are configured to be opened by Terminal, double click them in Finder. Otherwise, open Terminal then type /path/to/RunCubeDemo.sh and press Enter.

3. Method 3: Run the executable in Terminal.
   a. The current working directory must be where the executables are located! Use the `cd` (change directory) command from $HOME to `../path/to/CubeDemo/cube`. In Method 2, this detail is taken care of by the shell script.



Running Cube on Mac OS X Yosemite

To allow the Cube Demo to run on Mac OS X Yosemite, users need to manually register the installed SDL by following this procedure:

1. **Run the following command in Terminal:**`sudo find /path/to/SDL2.framework/Versions -name "SDL2" -exec codesign -f -s - {} \;`

   Replace the above path to SDL2 with your own setup. SDL2 is usually installed under either `~/Library/Frameworks` or `/Library/Frameworks`. When prompted, input your administrator account password and hit Enter.
2. Restart Xcode and rebuild the Cube Demo.

For more information, please reference http://www.paradiso.cc/2015/04/18/using-sdlv2-with-osx-yosemite/ .

On Windows

1. Method 1: Use the BAT scripts.
   a. Whether using the Wwise Launcher or the previous Wwise installer, the Wwise installation creates BAT files in the "<WwiseRoot>\Cube\cube" folder.
   b. Double click the BAT of the Cube Demo version you wish to run: debug, profiler, or release mode. Alternatively, open the Command Prompt and type the path to the BAT and press Enter.
2. Method 2: Run the executable from the Command Prompt.
   a. Navigate to the executable, "cube.exe", and run it. It can found under "<WwiseRoot>\Cube\Win32\<mode>\bin" where "<mode>" corresponds to the appropriate mode folder, namely Debug, Profile, or Release.

Using Other Wwise Documents and Support

The Wwise Cube Integration help is one component in our comprehensive set of materials and resources that are available for you. Consult the other Wwise documents as needed. For additional help, you can also visit our online support and resources center.

Chapter 2. Understanding Wwise's Sound Engine Integration into Cube

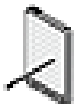
Cube, originally developed by Wouter van Oortmerssen, is an open source single or multi-player first person shooter game that is readily available from the Cube web site (http://cube.sourceforge.net/). To accommodate the integration of Wwise, the following modifications were made to the version downloaded from the Cube website:

- The existing sound code was rewritten to use the Wwise sound engine.
- The audio content was moved into a Wwise project.
- Dependencies on other libraries (SDL, SDL_image, libpng) were removed.
- Extra audio (footsteps, rocket thrust sound) was added.
- Some game config and content files were modified to accommodate the above changes.

About the Audio Integration Process

The core of the audio integration code is located in the following locations:

- Sound.cpp—includes all code that accesses the sound engine.
- Sound section of Protos.h—includes the declared public functions.



Note

These files can be found in the following folder: <WwiseRoot>/Cube Demo/cube_source/src.

Integrating the Wwise Sound Engine

The Wwise sound engine handles different types of game objects as well as all the game audio. In addition, several new commands were created to help you manage the sounds.

The following tasks were involved when integrating the Wwise sound engine into Cube:

- Registering Game Objects
- Integrating Audio
- Adding New Scripting Commands

Registering Game Objects

Two kinds of game objects were registered in the sound engine:

- Pointers to the baseent struct. These are the moving entities of the game world: players, monsters, and projectiles.

- Dummy game objects, numbered 64-127. These are used for 3D sounds that are not attached to a moving entity (for example, items spawning).

Integrating Audio

The following types of sounds were involved in the integration process:

Monster Play Sounds

- Each character has associated pain, die, and footstep events. (See monstertypes[] in monster.cpp.)
- Footstep events are posted from the monsterfootstep() function, which also sets the 'Material' switch corresponding to the texture underneath the character, to drive the footstep switch container.
- Items have associated pickup events. (See itemstats[] in entities.cpp)

Weapon Sounds

- Each weapon has an associated event. (See guns[] in weapon.cpp)
- Projectiles have a splashing event (S_FEXPLODE, S_RLHIT).
- The rocket has a continuous thrust sound (played by the event S_RLTHRUST). Velocity in baseent is boosted to enhance Doppler effect. The sound is stopped by the rocket splash event (S_RLHIT).
- Shooting while quad damage is enabled also posts the event S_ITEMPUP.

Network Sounds

- Certain events also need to go to the server for network play; this is handled by snd_clientevent().

Miscellaneous Sounds

- To find all other sound events occurring during game play, search for snd_event() calls in the code base.

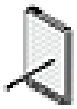
Adding New Scripting Commands

The following new sound-related commands can be used in the '.cfg' script files:

- akevent <string>: post an event (by name) on the local player game object (player1).
- soundvol <int>: to set the sound volume (0-255).
- musicvol <int>: to set the music volume (0-255).
- voicevol <int>: to set the voice volume (0-255).
- texturematerial <int> <string1> <string2>: this is essentially the same command as 'texture', but with an added parameter which is the associated material for footsteps.

Chapter 3. The Wwise Project

A Wwise project was created to facilitate the integration of the Wwise sound engine into Cube. Most of the sounds in the Audiokinetic version of Cube have been modified from the original game.



Note

The Wwise project file is located in the following folder: <WwiseRoot>/Cube Demo/WwiseProject.

The following sections describe some of the work that was done in the Wwise project.

Building the Actor-Mixer Hierarchy

The Actor-Mixer Hierarchy is where sound assets are organized in the project.

For the Cube project, the following audio structure was created:

- In the "Main" folder, four actor-mixers were created to group sounds into logical categories: Characters, Items, Monsters, and Weapons.
- Within the actor-mixers, a series of random and switch containers were used to add variety and distinctiveness to the sounds in the project.
- Randomizers were applied to the volume and pitch properties for some weapons and footsteps. The Randomizer randomly modifies the property values of the object each time it is played.
- All sounds within the "Main" folder are 3D sounds that use Game-defined positioning with a standard distance-based attenuation on volume.
- In the "Maps" folder, there are sounds, voices, and music specifically related to the map "dcp_the_core". All voices and music within this map are 2D sounds that are streamed from the hard drive.

Building the Master-Mixer Hierarchy

The Master-Mixer Hierarchy is a separate hierarchical structure of busses and auxiliary busses that allows you to group the many different sound and music structures within your project and prepare them for output.

For the Cube project, the following bus structure was created:

- Three main control busses were created: Music, SFX, and Voice.
- The Environments bus groups a series of auxiliary busses. Each auxiliary bus has a reverb effect inserted that represents a room in the game. Which auxiliary bus is processing at runtime is decided in the game's code.

- Auto-ducking was applied to the Voice bus so that Music is ducked when a Voice is played.
- An RTPC was applied to the Volume property of the three main control busses. These volume controls were mapped to the in-game volume faders (using game console).

Using Game Syncs

In Wwise, game syncs are used to efficiently manage specific changes in audio that relate to changes in action or conditions within the game. In the Cube project, the following game syncs were used:

- **Switches** - A switch group called "Material" was created to manage the different ground textures that exist in the game. Switches were created within this group for each ground texture, such as concrete and grass. These switches are used to define different footstep sounds for the main character as the surfaces change within the game.
- **Game Parameters** - Three game parameters were created that correspond to the volume faders within the Cube game. These game parameters were mapped to the bus volume property of the three main busses using an RTPC. By mapping the faders to the volume properties, game players can control the volume of sounds and music in the game themselves.

  The game parameter "PlayerHealth" is attached the the player's actual health gage. No sounds are using it by default, it's for you to attach this game parameter to sound object properties if you like.

  The Teleport looping Synth One sound uses the game parameter "Distance_to_Object" on the "Base Frequency" parameter of the synth to create an interesting pitch bend effect as you approach the teleporter. This game parameter can be reused on other sound objects.

Creating Events

Wwise uses events to drive the sound, music, and dialogue in-game. Events contain one or more actions that are applied to the different sound or music structures within your project hierarchy. For the Cube project, events were created and given names that matched the sound names triggered by Cube.

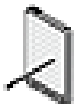
Generating SoundBanks

SoundBanks are basically the product of all your work and are the final audio package that becomes part of your game. They contain a group of events, Wwise objects, and/or converted audio files that will be loaded into a game's platform memory at a particular point in the game. For the Cube project, the following two SoundBanks were created:

- **Main** - Groups all events used in all maps.
- **dcp_the_core** - Groups the events for the map "dcp_the_core".

Fine-Tuning your Audio in Real Time in Game

After connecting to the Cube game, you can fine-tune your audio by modifying the properties for each sound in real time as the game is being played.



Note

Refer to the documentation to learn how to use the Remote Connection to connect to the game.

To modify sound properties in real-time (after Wwise has been connected to the game):

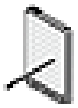
1. In Cube, fire the shotgun to listen to its current sound.
2. In Wwise, select the Shotgun sound object to load it into the Property Editor and Transport Control.

   The Shotgun sound object is located under the Weapons actor-mixer in the Main folder.
3. Modify the volume and pitch properties of the shotgun sound.
4. Click Play to listen to the new sound.
5. Go back to Cube and fire the shotgun again.

   You will notice that the new sound is automatically played in the game.
6. Experiment with other sounds in your game to see how quickly and easily you can fine-tune your game audio.

Integrating Changes into the Game

Although you can hear the changes you make while connected to the game, these changes are only temporary. To integrate these changes into your game, you must regenerate and replace the existing SoundBanks. After the SoundBanks have been replaced, the new sounds will be integrated into the Cube game.



Note

Make sure to turn Cube off before generating the new SoundBanks. When the game is on, one or more of the SoundBanks may be locked by the game preventing you from properly overwriting the SoundBank on the disk.

To generate new SoundBanks for Cube:

1. Switch to the SoundBank layout by doing one of the following:
   - From the menu bar, click **Layouts** > **SoundBank**.
   - Press **F7**.

   Notice that you have two SoundBanks for this project: DCP_the_core and Main.
2. From the SoundBanks list, click **Select All**.
3. From the Platforms list, select the **Windows** or **Mac** option depending on the operating system you work on.
4. From the Languages list, select the **English** (**US**) option.
5. From the menu bar, click **Project** > **Project Settings**.

   The Project Settings dialog box opens.
6. Switch to the **SoundBanks** tab.
7. In the Post-Generation Step group box, make sure that there is a command line for copying the streamed files in your project to the location where the SoundBanks are saved. By default, all projects have this command line as a post-generation step.
8. Click **OK** to close the Project Settings dialog box.
9. Click **Generate Selected** to begin generating the SoundBanks.

   The Generating SoundBanks dialog box opens where you can view the progress of the SoundBank generation process. When the SoundBanks have been generated, the SoundBanks Generation - Completed dialog box opens.
10. Click **Close**.

    The new SoundBanks have been generated and integrated into the Cube game. The next time you play Cube, the new sounds will be part of the game.

Known Issues

As this is just an example of how to integrate the Wwise sound engine into a game engine, the following issues were not resolved in this release.

Known SDL Migration Issues

- Full-screen is not available.
- Gamma command is disabled.
- Cannot see server messages when running dedicated.
- Mouse cursor 'escapes' from game when frame rate is low.
- Mouse wheel weapon selection is broken.

Known Sound Issues

- There are no footstep sounds for occluded characters.
- Footsteps can be heard in the water.
- There is no interactive music.

Chapter 4. Need Help?

Using the Help

The Wwise Help contains detailed information on each interface element in Wwise.

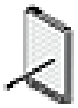
To open the Help from within Wwise, do one of the following:

- Click the Help icon (?) in the title bar of any of the views or dialog boxes.
- From the menu bar, click Help > Wwise Help.
- Press F1.

Further Resources

The Cube game and its Wwise project provide an idea of the workflow of developing a full game with Wwise. If you want to learn more about Wwise by looking at different approaches on game audio implementation, we suggest you take a look at the following resources:

- **Limbo** - Also available from the Wwise Launcher on a Windows, you can install Limbo, the multi-award game developed by Playdead with the full Wwise project. Limbo might be more suited for more advanced Wwise users because some sound hierarchies might appear quite complex to newcomers. Limbo is not available to install from the Wwise Launcher on a Mac.
- **Video Tutorials** - Video tutorials are available for all users in the community section of our website. They are also available on AudiokineticWwise's YouTube channel.
- **Wwise Certification** - The Wwise Certification online courses offer several programs for learning Wwise fundamentals and advanced specialized topics. Note that the Wwise-101 and Wwise-201 Certification Programs use Cube as the platform of choice for teaching Wwise.



Note

Materials sent by content providers were all originally high quality (48 kHz, 16 or 24 bits). However, to save download time and disk space, the size of some of the sound assets has been reduced.